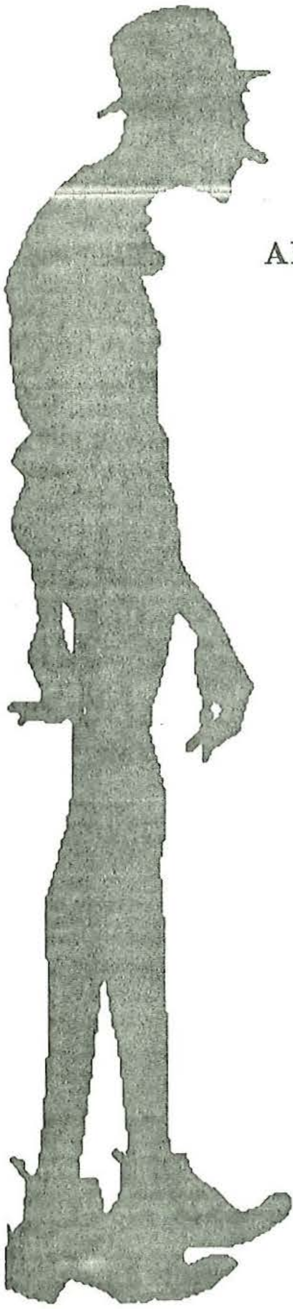
FORSCHUNGSBERICHTE
KÜNSTLICHE INTELLIGENZ



# ADAPTIVE CONFIDENCE AND ADAPTIVE CURIOSITY

Jürgen Schmidhuber

Report FKI-149-91

April 1991

TUM

TECHNISCHE UNIVERSITÄT MÜNCHEN
Institut für Informatik, Arcisstr. 21, 8000 München 2, Germany

# ADAPTIVE CONFIDENCE AND ADAPTIVE CURIOSITY

Jürgen Schmidhuber
Institut für Informatik
Technische Universität München
Arcisstr. 21, 8000 München 2, Germany
schmidhu@informatik.tu-muenchen.de

Technical Report FKI-149-91, April, 30, 1991

**Abstract**

Much of the recent research on adaptive neuro-control and reinforcement learning focusses on systems with adaptive 'world models'. Previous approaches, however, do not address the problem of modelling the reliability of the world model's predictions in uncertain environments. Furthermore, with previous approaches usually some ad-hoc method (like random search) is used to train the world model to predict future environmental inputs from previous inputs and control outputs of the system. This paper introduces ways for modelling the reliability of the outputs of adaptive predictors, and it describes more sophisticated and sometimes more efficient methods for their adaptive construction by on-line state space exploration: For instance, a 4-network reinforcement learning system is described which tries to maximize *the expectation of the temporal derivative of the adaptive assumed reliability of future predictions*. The system is 'curious' in the sense that it actively tries to provoke situations for which it *learned to expect to learn* something about the environment. An experiment with an artificial non-deterministic environment demonstrates that the method can be faster than the conventional model-building strategy.

## 1 THE PROBLEM

Much of the recent research on adaptive neuro-control and reinforcement learning focusses on systems with sub-modules that learn to predict inputs from the environment. These sub-modules often are called 'adaptive world models'; they are useful for a whole variety of control tasks. For instance, Werbos' and Jordan's architectures for neuro-control [16][3] contain an adaptive world model in form of a back-propagation module (the model network) which is trained to predict the next input, given the current input and the current output of an adaptive control network. The model network allows to compute error gradients for the controller outputs. This is essential, since with typical adaptive neuro-control tasks there is no teacher who provides desired controller outputs. There is only a desired environmental input. Extensions of this approach (e.g. [11]) rely on the same basic principles. Sutton's 'DYNA-systems' [13] use adaptive world models for limiting the number of 'real-world experiences' necessary to solve certain reinforcement learning tasks.

There are at least two important problems with all of these approaches that have not been addressed so far:

*1. Previous model-building control systems are not well-suited for uncertain non-deterministic environments.* In particular, they do not model the reliability of the predictions of the adaptive world models. Therefore, if credit assignment for the controller is based on the assumption of a correct world model, unexpected results may be obtained.

*2. Previous model-building control systems employ some ad-hoc method for establishing the world model.* For instance, Jordan [3], Jameson [2], Nguyen and Widrow [5], Schmidhuber and Huber [12], and others use random search to train the world model. Sutton [13] uses a local input/output representation and makes the probability of making a certain training experiment dependent on the time that went by since the system made the last experiment of the same type. These methods work fine for certain problems, but they do not address the challenges of real world tasks in uncertain environments. There are at least two (related) sources of efficiency which are neglected by these approaches:

*2A. Not much additional training time should be wasted on exploring those parts of the world which are already well-modelled. 2B. Not much additional training time should be wasted on exploring those parts of the world where the expectation of future improvement of the world model is low.*

The first contribution of this paper (section 2) is to show how one can adaptively model the reliability of a predictor's predictions, and how adaptive controllers can profit from this additional knowledge.

The second (and most important) contribution of this paper (section 3) is to show how reinforcement learning can be used for teaching a model-building controller to actively generate training examples for increasing the reliability of the predictions of its world model. This is relevant for the problem of 'on-line state space exploration'. The approach is based on learning to estimate the effects of further learning.

## 2 ADAPTIVE CONFIDENCE

Consider an adaptive discrete time 'predictor' $M$ (not necessarily a neural network) whose input at time $t$ is the real vector $i_M(t)$ and whose output at time $t$ is the real vector $o_M(t) = f_M(i_M(t), h_M(t))$, where the real vector $h_M(t)$ represents the internal state of $M$. Meaningful internal states are required if the prediction task requires to memorize past events. At time $t$ there is a target output $d_M(t)$. The predictor's goal is to make $o_M(t) = d_M(t)$ for all $t$.

After having provided a number of training examples for $M$, $M$ usually will still make some errors, particularily if the training environment is noisy. How can we model the reliability of $M$'s predictions?

We introduce an additional 'confidence module' $C$ (not necessarily a neural network) whose input at time $t$ is the real vector $i_C(t) = i_M(t)$ and whose output at time $t$ is the real vector $o_C(t) = f_C(i_C(t), h_C(t))$, where the real vector $h_C(t)$ is the internal state of $C$. At time $t$ there is a target output $d_C(t)$ for the confidence module. $d_C(t)$ should provide information about how reliable $M$'s prediction $o_M(t)$ can be expected to be [10] [8] [9].

In what follows, $v^j$ is the $j$th component of a vector $v$, $E$ denotes the expectation operator, $dim(x)$ denotes the dimensionality of vector $x$, $\mid c \mid$ denotes the absolute value of scalar $c$, $P(A \mid B)$ denotes the conditional probability of $A$ given $B$, and $E(A \mid B)$ denotes the conditional expectation of $A$ given $B$. For simplicity, we will concentrate on the case of $h_C(t) = h_M(t) = 0$ for all $t$. This means that $M$'s and $C$'s current outputs are based only on the current input. There is a variety of simple ways of representing reliability in $d_C(t)$:

*1. Modelling probabilities of global prediction failures.* Let $d_C(t)$ be one-dimensional. Let $d_C(t) = P(o_M(t) \neq d_M(t) \mid i_M(t))$. $d_C(t)$ can be estimated by $\frac{n_1}{n_2}$, where $n_2$ is the number of those times $k \leq t$ with $i_M(k) = i_M(t)$ and where $n_1$ is the number of those times $k$ with $i_M(k) = i_M(t), o_M(k) \neq d_M(k)$.

*2. Modelling probabilities of local prediction failures.* Let $d_C(t)$ be $dim(d_M(t))$-dimensional. Let $d_C^j(t) = P(o_M^j(t) \neq d_M^j(t) \mid i_M(t))$ for all appropriate $j$. $d_C^j(t)$ can be estimated by $\frac{n_1}{n_2}$, where $n_2$ is the number of those times $k \leq t$ with $i_M(k) = i_M(t)$ and where $n_1$ is the number of those times $k$ with $i_M(k) = i_M(t), o_M^j(k) \neq d_M^j(k)$.

Variations of method 1 and method 2 would not measure the probabilities of exact matches between predictions and reality but the probability of 'near-matches' within a certain (e.g. euclidian) tolerance.

*3. Modelling global expected error.* Let $d_C(t)$ be one-dimensional. Let

$$d_C(t) = E\left\{\frac{1}{2}(d_M(t) - o_M(t))^T(d_M(t) - o_M(t)) \mid i_M(t)\right\}.$$

If $C$ is a back-propagation net, an approximation of $d_C(t)$ can be obtained by using gradient descent (with a small learning rate) for training $C$ at time $t$ to emit $M$'s error $\frac{1}{2}(d_M(t) - o_M(t))^T(d_M(t) - o_M(t))$. This is a special case of the method described in [10] (there a fully recurrent net was employed). Of course, other error functions are possible. For instance, with the experiments described below the confidence network predicted the the absolute value of the difference between $M$'s (one-dimensional) output and the current target value.

*4. Modelling local expected error.* Let $d_C(t)$ be $dim(d_M(t))$-dimensional. Let

$$d_C^j(t) = E\{(d_M^j(t) - o_M^j(t))^2 \mid i_M(t)\}$$

for all appropriate $j$. If $C$ is a back-propagation net, an approximation of $d_C(t)$ can be obtained by using gradient descent (with a small learning rate) for training $C$ at time $t$ to emit $M$'s local prediction errors

$$\left((d_M^1(t) - o_M^1(t))^2, \ldots, (d_M^m(t) - o_M^m(t))^2\right)^T,$$

where $m = dim(o_M(t))$.

If $M$ is used as a 'world model', then with many applications $i_M(t) = o_A(t) \circ x(t)$ and $d_M(t) = x(t+1)$, where $o_A(t)$ is the output vector of a controller $A$ at time $t$, '$\circ$' is the concatenation operator, and $x(t)$ is the environmental input at time $t$. In general, $o_A(t)$ influences the state of the environment. Therefore it may have an influence on $x(t+1)$.

In [9] confidence modules have been successfully applied to the problem of meaningful hierarchical sequence chunking. The next subsection describes how they can be used for supporting controller learning. The next subsection is not essential for the central contribution of this paper (which can be found in section 3).

## 2.1 USING CONFIDENCE MODULES FOR SUPPORTING CONTROLLER LEARNING

The general principle behind the ideas described below is: Any adaptive controller whose credit assignment process is based on a world model should be modified only if the predictor's predictions are reliable.

Among the most radical implementations of this principle are the ones described next. These are independent of the precise nature of the credit assignment process of the controller:

*1. If method 1 or method 3 of section 2 is employed, use the model for training the controller at time t only if* $d_C(t) \leq \beta = const. \geq 0$.

*2. If method 2 or method 4 is employed, use* $o_M^j(t)$ *for training the controller at time t only if* $d_C^j(t) \leq \beta = const. \geq 0$.

For instance, let us assume that $M$ is used to approximate

$$\frac{\partial[(x(t+1) - y(t+1))^T (x(t+1) - y(t+1))]}{\partial o_A(t)}$$

by

$$\frac{\partial[(o_M(t) - y(t+1))^T (o_M(t) - y(t+1))]}{\partial o_A(t)},$$

where $y(t)$ is a desired state of the environment at time $t$. This is a common principle of the approaches described in [16], [3] and [11], where the gradient computed with the help of $M$ is propagated down into the controller where it causes weight changes according to the rules of gradient descent.

If method 2 or method 4 is employed, compute only

$$\sum_{j:(o_M^j(t) - x^j(t+1))^2 \leq \beta,\, d_C^j(t) \leq \beta} \frac{\partial(o_M^j(t) - y^j(t+1))^2}{\partial o_A(t)},$$

and use this value for training the controller (here $\beta$ is a small non-negative constant).

A less radical strategy is to weight highly reliable predictions more heavily than less reliable ones. For instance, if method 2 or method 4 is employed, compute

$$\sum_j \sigma(c_O^j(t)) \frac{\partial (o_M^j(t) - y^j(t+1))^2}{\partial o_A(t)}$$

and use this value for training the controller. Here $\sigma$ is a monotonically decreasing function whose output values are in $[0, \ldots, 1]$.

The effect of all these strategies is that the controller is preferrably trained with respect to those predictions of $M$ that are assumed to be reliable.

## 3 ADAPTIVE CURIOSITY

This section provides the major contribution of this paper. We define curiosity as the desire to improve a predictor of the reactions of an environment (a 'world model'). In [10], [6] and [7] the following basic idea for 'on-line state space exploration by implementing dynamic curiosity and boredom' has been formulated: *Spend reinforcement for a model-building controller whenever there is a mismatch between the expectations of the adaptive world model and reality.* Any sensible reinforcement learning algorithm can be used to encourage the controller to generate action sequences that provoke situations where the world model tends to make bad predictions. Since the model is adaptive, its predictions often will improve. This in turn will lead to less reinforcement for the controller. Therefore the corresponding action sequences will become discouraged. The controller will get *'bored'* with the corresponding situations and will start to focus on yet unpredictable parts of the environment.

The particular implementation described in [10] and [7] employed a recurrent confidence network with a one-dimensional output for modelling the expected error of the model network (this error was called the 'curiosity reinforcement'). The confidence network was not called so: It was part of the model network (which predicted the next state of the environment plus a reinforcement vector including all kinds of reinforcement, not just 'curiosity reinforcement'). The target activation of the single output unit of the confidence net was a function of the current error of the model network. In the simplest case this function was linear. The controller's goal was to activate the error-predicting unit by creating action sequences for provoking mismatches between expectations and reality. The gradient computed for the error predictor also served to change the internal representations of the whole network (whose error function simply contained an additional term). Recently Thrun and Möller described related ideas [14] (they use the term 'competence network' instead of the term 'confidence network' as used in [9] and [8]).

One problem with the idea above is that in non-deterministic environments the controller will focus on parts of the environmental dynamics which are inherently unpredictable. This is because the adaptive model usually will produce incorrect predictions for the uncertain parts of the environment. Therefore the controller will receive reinforcement *although it cannot be expected that the world model will improve.*

A related problem is that often certain parts of the environment can be represented only by a complex mapping which is difficult to learn while other parts are 'easy to learn'. If we want a system which first tries to solve the easy tasks before focussing on the harder tasks then the system will need an (adaptive) internal representation of something like the *expectation of how difficult certain learning tasks will be.*

Both problems are related in the sense that both require to *learn something about the effects of further learning.* In what follows an approach for coping with these problems will be described. Instead of simply learning to predict errors as the approach described in [10] the new approach learns to predict cumulative error *changes.*

### 3.1 THE BASIC PRINCIPLE

This subsection discusses a rather general principle of adaptive curiosity. Here we do not have to care whether the adaptive world model is implemented as a back-propagation network, as a lookup table, or as something else. There are certain natural implementations of the ideas; they are discussed in the following subsections.

The basic principle can be formulated as follows: *Learn a mapping from actions (or action sequences) to the expectation of future performance improvement of the world model. Encourage action sequences where this expectation is high.*

One way to do this is the following (section 4 will describe alternatives): *Model the reliability of the predictions of the adaptive predictor as described in section 2. At time t, spend reinforcement for the model-building control system in proportion to the current change of reliability of the adaptive predictor. The 'curiosity goal' of the controller (it might have additional 'pre-wired' goals) is to maximize the expectation of the cumulative sum of future positive or negative changes in prediction reliability.*

More formally: The controller's curiosity goal at time $t_0$ is to maximize

$$E\{\sum_{t \geq t_0} -\gamma^{t-t_0} \triangle o_C(t+1)\}.$$

Here $0 \leq \gamma < 1$ is a discount factor for avoiding infinite sums, and $\triangle o_C(t)$ is the (positive or negative) change of assumed reliability caused by the observation of $i_M(t)$, $o_M(t)$, and $x(t+1)$.

For instance, if method 1 or method 3 from section 2 is employed, then $\triangle o_C(t) = o_C(t) - \bar{o_C}(t)$, where $\bar{o_C}(t)$ is $C$'s response to $i_M(t)$ after having adjusted $C$ at time $t$.

So far the discussion did not have to refer to a particular reinforcement learning algorithm. Every sensible reinforcement learning algorithm ought to be useful (e.g [1][17][15][11]). The following subsections focus on two particular methods, namely, adaptive critics and Q-learning.

## 3.2 CRITIC-BASED MODEL-BUILDING CONTROLLERS FOR ON-LINE STATE SPACE EXPLORATION

A straight-forward way to build a 'curious' model-building control system based on the principle described above is the following:

Use a controller $A$, a predictor $M$ (the world model) and a confidence module $C$ as described in section 2. In addition, train an adaptive critic [1][16][15] $P$ whose output at time $t$ is $o_P(t)$ to predict

$$\sum_{t \geq t_0} -\gamma^{t-t_0} \triangle o_C(t+1)$$

at time $t_0$ by training it to predict

$$\triangle o_C(t_0+1) + \gamma o_P(t_0+1).$$

Use

$$r(t) = \triangle o_C(t+1) + \gamma o_P(t+1) - o_P(t)$$

as reinforcement for the reinforcement learning algorithm of the controller at time $t$.

The method considers only 'curiosity reinforcement'. If there is additional external reinforcement, then the adaptive critic simply will have to be trained to predict the (discounted) sum of curiosity reinforcement and external reinforcement [7][10].

## 3.3 A CURIOUS SYSTEM BASED ON Q-LEARNING

Here we describe how a reinforcement learning method called Q-learning can be used to build a 'curious' model builder. The notation is the same as above. Following [15] we introduce an adaptive function $Q$ for evaluating pairs of inputs $x(t)$ and actions $a(t)$ as well as an utility function $U$ for evaluating inputs $x(t)$.

After random initialization of $C$, $M$, $A$, $U$, and $Q$, at each time step $t$ the following algorithm is performed:

1. Randomly select $p \in [0, \ldots, 1]$. If $p \leq \mu \in [0, \ldots, 1]$ then $a(t) = o_A(t)$ else $a(t)$ is chosen randomly.
2. Compute $o_M(t)$, execute $a(t)$, obtain $x(t+1)$, and adjust $M$ to improve its prediction in similar situations. Adjust $C$ according to one of the methods described in section 2. Obtain $r(t) = r_{ext}(t) + \triangle o_C(t)$, where $r_{ext}(t)$ is the current externally defined reinforcement (if there is any) and where $\triangle o_C(t)$ is the current change of confidence in $M$'s current predictions.
3. Set $Q(x(t), a(t)) \leftarrow (1-\alpha)Q(x(t), a(t)) + \alpha(r(t) + \gamma U(x(t+1)) - Q(x(t), a(t)))$, where $\alpha$ is a learning rate and $0 \leq \gamma \leq 1$.
4. Adjust $A$ to emit $a$ in response to $x(t)$ such that $Q(x(t), a) = max_b Q(x(t), b)$.
5. $U(x(t)) \leftarrow Q(x(t), a)$.

Note that the algorithm does not specify the implementation of $C$, $M$, and $A$. All three can be implemented as lookup tables or (in hope for useful 'generalizations') as back-propagation networks, Boltzmann-machines, etc. $Q$ and $U$ may be replaced by back-propagation networks, too (see the experiments described in section 5).

## 4 PREDICTING ERROR CHANGES DIRECTLY

The reinforcement generating mechanism for the reinforcement learning systems described above can be modified in various ways. For instance, define $\bar{o_M}(t)$ as $M$'s response to $i_M(t)$ *after* having adjusted $M$ at time $t$. We can replace the confidence network by a network $H$ which at every time step receives the current input $i_M(t)$ and whose target output is the current *change* of $M$'s output $\triangle o_M(t) = o_M(t) - \bar{o_M}(t)$ caused by $M$'s learning algorithm ($H$ should have a small learning rate). $H$ will learn approximations of the expectations

$$E\{\triangle o_M(t) \mid i_M(t)\}$$

of the changes of $M$'s responses to given inputs. The *absolute value* $\mid o_H(t) \mid$ of $H$'s output $o_H(t)$ (an approximation of $\mid E\{\triangle o_M(t) \mid i_M(t)\} \mid$) should be taken as the reinforcement for the adaptive critic or the Q-learning algorithm (the reinforcement learning algorithm does not have to be specified here): The controller's curiosity goal at time $t_0$ is to maximize

$$E\{\sum_{t \geq t_0} -\gamma^{t-t_0} \mid o_H(t) \mid\},$$

where $0 \leq \gamma < 1$ is a discount rate.

An alternative would be to make predictions about the (discounted) sum of future changes of $M$'s *weight vector* and use these predictions in an analoguous manner.

## 5 EXPERIMENTS

The following experiments with a method based on the 'curious' model-building Q-learner above were conducted by Reiner Wahnsiedler, a student at TUM.

An uncertain and only partly deterministic environment for an artificial agent was defined. It was a square consisting of 10 rows or columns with 10 slots each. At a given time step the agent occupied one out of the 100 slots. It was able to execute one out of 4 possible actions: 1. Move to the slot to the left of the current slot if such a slot exists. Otherwise move to the rightmost slot of the same row. 2. Move to the slot above the current slot if such a slot exists. Otherwise move to the slot at the bottom of the same column. 3. Move to the slot below the current slot if such a slot exists. Otherwise move to the slot at the top of the same column. 4. Move to the slot to the right of the current slot if such a slot exists. Otherwise move to the leftmost slot of the same row.

A predictor network $M$, a confidence module $C$ and a module for evaluating pairs of positions and actions $Q$ were implemented as general back-propagation networks. With these particular experiments, however, the networks had no hidden layers. Therefore the networks degenerated to look-up tables, due to the local input representation to be described below.

At time $t$ the agent's input $i(t)$ was a representation of the current position. Each possible position was represented by a 100-dimensional bitvector with only one non-zero component (local input representation). Whenever the agent entered a slot it observed one out of two possible 'reactions' $s(t) \in \{0, 1\}$. There were three classes of slots: With members of class 1 $s(t)$ was always 1. With members of class 2 $s(t)$ was always 0. With members of class 3 $s(t)$ was selected randomly; the probability that the reaction was 1 was equal to $\frac{1}{2}$.

Between two successive time steps $t-1$ and $t$ the following operations were executed: At time $t-1$ $Q$ received four different 400-dimensional bit-vectors with only one non-zero component as inputs (local input representation). There was one input vector for each combination of $i(t-1)$ and the four possible actions. $Q$ emitted a one-dimensional output in response to each of the four inputs [4]. These four outputs were interpreted as evaluations of the corresponding input/action pairs. $Q$'s output unit employed the logistic activation function $\frac{\frac{6}{10}}{1+e^{-zq}} - \frac{1}{10}$, where $z$ was the current weight of the connection from $Q$'s currently active input unit to the output unit and $q$ was the current activation of this input unit. The controller was not implemented as a separate module: It was implemented as the simple mechanism which chose the action corresponding to the highest of the four current evaluations of $Q$. This action was executed to obtain $i(t)$. $Q$'s input vector corresponding to the chosen action was called $i_Q(t-1)$.

Then, at time $t$, $M$ received $i(t)$ as an input and produced a one-dimensional output $o_M(t)$. With this particular experiment $M$'s task was not to learn to predict $i(t+1)$ but to predict $s(t)$. The apparently rather simple task is sufficient to illustrate the basic problem: Obviously only slots from class 1 or class 2 allowed to learn 'good' predictions. But, the system was not told in advance which parts of the environment were deterministic and which were not.

$M$'s output unit employed the activation function $\frac{2}{1+e^{-wm}} - \frac{1}{2}$, where $w$ was the current weight of the connection from $M$'s currently active input unit to the output unit and $m$ was the current activation of this input unit.

At time $t$ the confidence network $C$ received $i(t)$ as an input. Its one-dimensional output $o_C(t)$ was interpreted as a prediction of $E\{| o_M(t) - s(t) || i(t)\}$ (a variant of method 3 of section 2). $C$'s output unit employed the activation function $\frac{2}{1+e^{-vc}} - \frac{1}{2}$, where $v$ was the current weight of the connection from $C$'s currently active input unit to the output unit and $c$ was the current activation of this input unit. To obtain the current increase or decrease in confidence $\triangle o_C(t)$ the following procedure was employed: After having computed $o_C(t)$, $| o_M(t) - s(t) |$ served as the desired output for $C$. $C$'s weights were adjusted according to the rules of gradient descent. Then $o_{\bar{C}}(t)$ ($C$'s response to $i(t)$ *after* the current weight modification, see section 3.1) was computed.

$\triangle o_C(t) = o_C(t) - o_{\bar{C}}(t)$ served as the current 'curiosity reinforcement' (there was no external reinforcement besides the 'curiosity reinforcement'). $Q$ was updated according to the rules of gradient descent: At time $t-1$, $Q$'s *desired target output* in response to $i_Q(t-1)$ was $\triangle o_C(t) + \gamma U(t)$, where $U(t)$ was the evaluation of the best action proposed by $Q$ at time $t$.

With one experiment the world contained a 3 times 3 slots subsection of 9 slots which always produced deterministic reactions. The other 91 slots always produced random reactions. The 'curious' system was tested against the conventional random search method. In both cases $M$'s learning rate was 0.1, and all weights were initialized between -0.1 and 0.1. With the 'curious' system, in the beginning of the training phase the agent was placed at a randomly chosen position. $C$'s learning rate was 1.0, the learning rate of the $Q$-function was 40.0, and $\gamma$ was 0.9. Actions were chosen deterministically (this corresponds to $\mu = 1$ in the algorithm of section 3.3). With both methods, at time $t$ the sum $E(t)$ of the squared differences between the values of the reactions of the deterministic slots and the corresponding predictions of $M$ was used as a criterion for judging the quality of $M$.

*It turned out that with guidance by the principle of adaptive curiosity $E(t)$ decreased clearly faster than with random search.* For instance, with random search about 2700 training examples for $M$ were needed to make $E(t)$ smaller than 0.3. With the 'curious' system only about 1000 training examples were needed to make $E(t)$ smaller than 0.15. Similar results were obtained with the strategy described in section 4.

Of course, the reason for this superior performance was the following one: The 'curious' system soon detected that there were certain slots where further improvement of the world model could be expected. *It started to focus on these particular slots.* The random search method was not selective at all, therefore it wasted a lot of time on senseless exploration of parts of the environment that did not allow performance improvement. (Note that the method described in [10] will perform even worse in this partly non-deterministic environment: Soon it will prefer to lead the agent to the non-deterministic slots, simply because there the expected error is highest.)

The task described above certainly is a toy problem. But, the more complex the environment the more benefits should be expected from the principle of adaptive curiosity. Ongoing experiments focus on increasingly complex worlds, non-local input/output representations and on the expected 'generalization capabilities' of non-trivial networks with hidden units.

## 6 CONCLUSION

The central idea of this paper is to construct an adaptive system which *learns to predict the effects of further learning.* This is done by training an adaptive sub-module to predict (the expectation of the sum of) future error *changes* caused by a particular learning algorithm. Here one adaptive module learns to make estimates about the effects of the learning procedure of another adaptive module. In other words, there is a module which learns to make a statement about learning itself. This is related to the concept of *'meta-learning'* briefly touched upon in a different context in [7]. In a very limited sense the system learns how to learn.

The method represents a general strategy for learning to select training examples such that the expected performance improvement is maximized. Therefore the usefulness of the approach is not limited to model-building control systems. The principles above are general enough to be of interest whenever the task is to select appropriate training examples for any kind of learning system.

## 7 ACKNOWLEDGEMENTS

I would like to thank Reiner Wahnsiedler who conducted the experiments.

## References

[1] A. G. Barto, R. S. Sutton, and C. W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13:834–846, 1983.

[2] J. Jameson. A neurocontroller based on model feedback and the adaptive heuristic critic. In *Proc. IEEE/INNS International Joint Conference on Neural Networks, San Diego*, volume 2, pages 37–43, 1990.

[3] M. I. Jordan and D. E. Rumelhart. Supervised learning with a distal teacher. Technical Report Occasional Paper #40, Center for Cog. Sci., Massachusetts Institute of Technology, 1990.

[4] L. Lin. Self-improving reactive agents: Case studies of reinforcement learning frameworks. In J. A. Meyer and S. W. Wilson, editors, *Proc. of the International Conference on Simulation of Adaptive Behavior: From Animals to Animats*, pages 297–305. MIT Press/Bradford Books, 1991.

[5] Nguyen and B. Widrow. The truck backer-upper: An example of self learning in neural networks. In *IEEE/INNS International Joint Conference on Neural Networks, Washington, D.C.*, volume 1, pages 357–364, 1989.

[6] J. H. Schmidhuber. Dynamische neuronale Netze und das fundamentale raumzeitliche Lernproblem. Dissertation, Institut für Informatik, Technische Universität München, 1990.

[7] J. H. Schmidhuber. Making the world differentiable: On using fully recurrent self-supervised neural networks for dynamic reinforcement learning and planning in non-stationary environments. Technical Report FKI-126-90 (revised), Institut für Informatik, Technische Universität München, November 1990. (Revised and extended version of an earlier report from February.).

[8] J. H. Schmidhuber. Talk at the NIPS'90 workshop on dynamic networks led by R. Rohwer, 1990.

[9] J. H. Schmidhuber. Adaptive decomposition of time. In O. Simula, editor, *Proceedings of the International Conference on Artificial Neural Networks ICANN 91, to appear*. Elsevier Science Publishers B.V., 1991.

[10] J. H. Schmidhuber. A possibility for implementing curiosity and boredom in model-building neural controllers. In J. A. Meyer and S. W. Wilson, editors, *Proc. of the International Conference on Simulation of Adaptive Behavior: From Animals to Animats*, pages 222–227. MIT Press/Bradford Books, 1991.

[11] J. H. Schmidhuber. Reinforcement learning in markovian and non-markovian environments. In D. Touretzky and D. S. Lippman, editors, *Advances in Neural Information Processing Systems 3, in press*. San Mateo, CA: Morgan Kaufmann, 1991.

[12] J. H. Schmidhuber and R. Huber. Learning to generate artificial fovea trajectories for target detection. *International Journal of Neural Systems, to appear*, 1991.

[13] R. S. Sutton. First results with DYNA, an integrated architecture for learning, planning and reacting. In *Proceedings of the AAAI Spring Symposium on Planning in Uncertain, Unpredictable, or Changing Environments*, 1990.

[14] S. Thrun and K. Möller. On planning and exploration in non-discrete environments. Technical report, Gesellschaft für Mathematik und Datenverarbeitung, D-5205 St. Augustin, Germany, March 1991.

[15] C. Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, 1989.

[16] P. J. Werbos. Building and understanding adaptive systems: A statistical/numerical approach to factory automation and brain research. *IEEE Transactions on Systems, Man, and Cybernetics*, 17, 1987.

[17] R. J. Williams. Toward a theory of reinforcement-learning connectionist systems. Technical Report NU-CCS-88-3, College of Comp. Sci., Northeastern University, Boston, MA, 1988.